CSE 3105 / CSE 3137

OBJECT-ORIENTED ANALYSIS AND DESIGN

FALL 2024

COURSE PROJECT: *LIBRARY MANAGEMENT SYSTEM*

Requirements Analysis Document

Group 19

Sıla Naz ASLAN – 210316072

Sena KILINÇ– 210316036

Aslı ÇELİK – 210316024

Umut AKBAŞ – 210316010

Zafer SARIK – 200316034

İsmail Ertuğrul BOZKURT – 210316059

10 December 2024

Table Of Contents

1 Introduction

The name of our project is the Library Management System. In our decision-making process for this project we addressed one of the problems experienced by students, the library system. In the current system since tables in the common area cannot be reserved except for individual rooms, students can not know whether the library is full or empty. This situation creates a time problem for students. If we organize this as we think, this problem will be eliminated. When students enter the library system they can see which table in the common area is empty, they can choose which table they will sit at and how many hours they will sit. Apart from these, hourly updates are also made. The purpose of this is to check whether the student is still there. With all these arrangements in place, the problems that currently arise will be resolved..

2 Current System

Let’s say students use the school library during exam periods or to prepare for projects. In the current system, there is no system that shows where the available tables are in the library. When students come to the library, they have to check every table to find an available table. Especially during busy times (for example, after classes are over), students who come to the library spend time looking for an available table. This process may limit the time students have to work on group projects or study. Students who cannot find an available table may have to leave the library or go somewhere else. The current system only allows students to book individual rooms; therefore, there is no reservation or occupancy control system for tables in the common study area in the library. When students come to the library, they cannot see if there are any available tables and have to walk around the entire area to find one. This wastes time, especially during busy times, and negatively affects students’ productive study hours. If the library could provide a table occupancy status and reservation system, students could see which tables are available before they come to the library and use their time more efficiently. Students have difficulty in managing library usage as the current system does not provide any occupancy tracking or table reservation solution to solve these problems. This hinders the efficient use of library resources and reduces student satisfaction.

3 Proposed System

3.1 Functional Requirements

Login: A user login module for students to log in to the system.

View Empty Tables: Students should be able to instantly see the empty and occupied tables in the library through the system.

Make a Table Reservation: Students should be able to reserve a table that they will use for a certain period of time from the system.

Update the Reservation Time: Students should be able to update the reservation time when they want to continue using the table.

Check Hourly Updates: The system should check whether the student is still at the table they reserved by making hourly updates.

Notify Reservation Expiration: A reminder or warning notification should be sent to students whose reservation period has expired.

3.2 Nonfunctional Requirements

Performance: The system should be able to update the status of the tables instantly and perform the user's transactions (login, reservation, update) quickly.

Reliability: The system should work without interruption and ensure that the operations performed by users are performed correctly. It should be able to make a quick turnaround in case of malfunctions or errors.

Usability: The system should have a user-friendly interface and allow users to find and reserve tables quickly and easily. It should be supported by visuals and assistance documents.

Supportability: The system should be manageable in terms of maintenance and updates, and users can get support for a Help Desk where the problems they face or the user documentation must be provided.

3.3 System Models

3.3.1 Scenarios

- ***The Table Reservation Process Scenario:***

When Zafer enters the library, he first goes to the login page of the Library Management System and logs in by entering his username and password by clicking on the "Login" button. After his entry is successful, he comes across a list on the main screen where he can see empty and full desks in the library. Zafer can instantly see the empty tables from this list and check which tables are full. Zafer chooses an empty desk that he considers suitable for study. Detailed information about the table (table number, location, etc.) after viewing, Zafer determines the amount of time he wants to sit. After setting this time limit, he completes his reservation by clicking on the “Make a Table Reservation” button. The system shows a confirmation message that the reservation was made successfully, and the status of the selected table is updated to "reserved".

After making his reservation, if Zafer wants to spend more time at the table later, he determines the new period to extend the existing booking period by clicking on the "Update the Reservation Time " option. The system also successfully performs this update and shows the new booking period. The system makes hourly updates every hour to check if Zafer’s booking period has expired. If Zafer has not left the table, the system checks him hourly and keeps the status of the table up to date.

Finally, when the booking period expires, the system sends Zafer a reminder notification that his booking has expired. Thanks to this, Zafer instantly finds out about the time he spent at his desk and that his reservation has expired. These steps positively affect the library experience of students by ensuring that the system works in a user-oriented and efficient way.

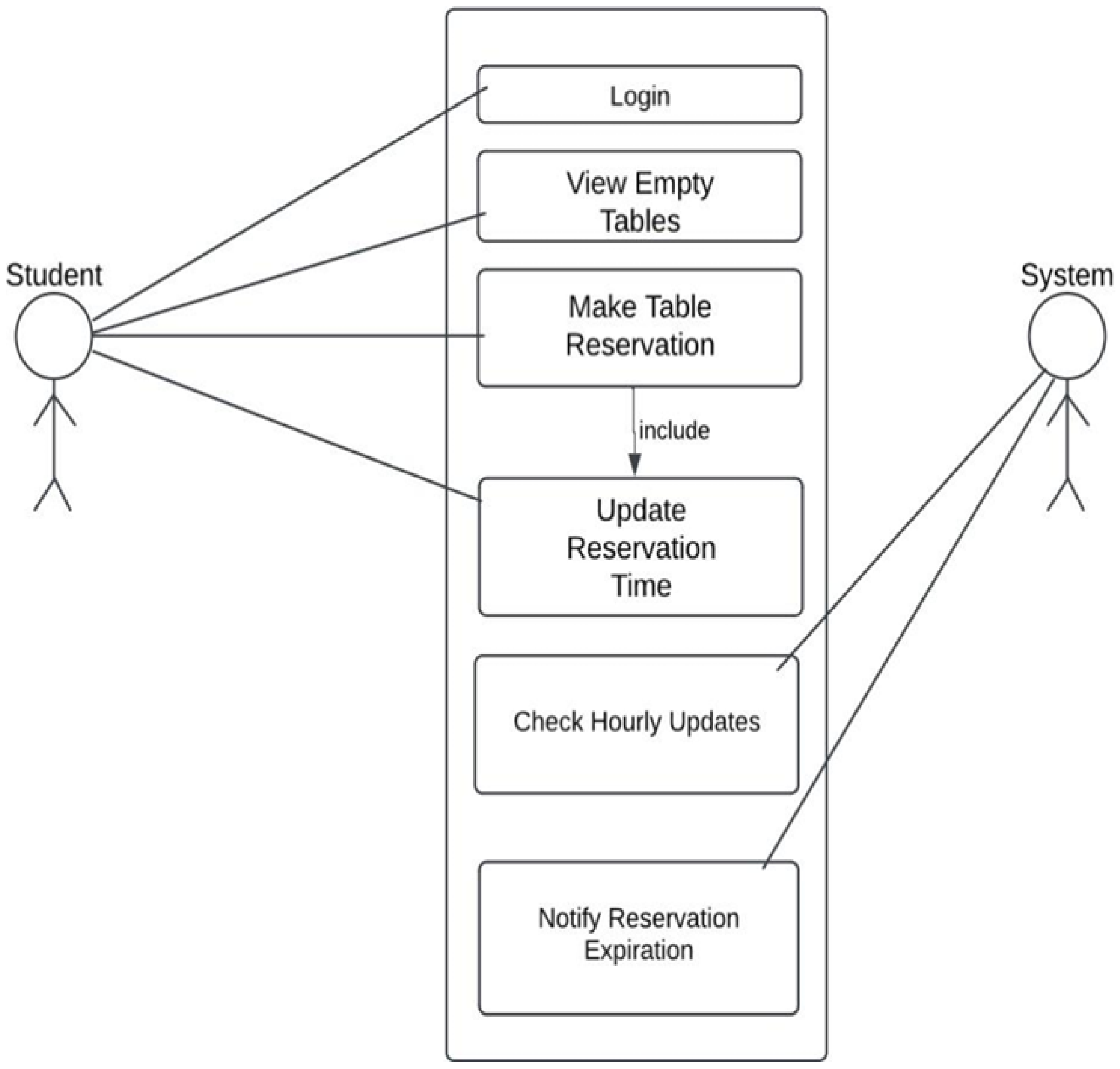
- ***Booked Table Warning Scenario:***

  When Sıla tries to choose a table on the table selection screen, if the table she wants to choose is full, the warning "This table is already reserved." appears on the screen. Sıla understands that she needs to choose another table and therefore does not waste time with tables that are already full.

- ***Update Reservation Scenario:***

  Ismail decides that he still wants to work at the table he previously reserved. He clicks on the “Update the Reservation Time” option on the main screen, determines the new duration to extend the current reservation period and clicks the “Update” button. The system successfully updates his reservation and displays the new duration.

3.3.2 Use Case Model



Use Case Name: ***The Table Reservation Process***
Participating Actors:

- ***User (Student):*** Initiates the reservation process and interacts with the system.
- ***System:*** Manages table availability, reservations, and notifications.

Flow of Events:

1. ***User logs in:***
   - The user accesses the library management system by entering their username and password on the login page.
2. ***View table status:***
   - After successful login, the system displays a list of available and occupied tables.
3. ***Select a table:***
   - The user selects an empty table suitable for their study session.
4. ***Set reservation time:***
   - The user specifies the desired reservation duration.
5. ***Confirm reservation:***
   - The user confirms the reservation by clicking the "Make a Table Reservation" button. The system updates the table's status to "reserved" and displays a confirmation message.
6. ***Update reservation time (if necessary):***
   - If the user decides to extend their stay, they can update the reservation time by selecting "Update the Reservation Time" and entering the new duration. The system processes the update and shows the new booking period.
7. ***Receive expiration notification:***
   - When the reservation period ends, the system sends a notification to remind the user that their booking has expired.

Entry Condition:

- The user must have an active account and be logged into the system.

Exit Conditions:

- The user logs out or the reservation process ends successfully.
- The reservation period expires, and the system notifies the user.

Use Case Name: ***Booked Table Warning***
Participating Actors:

- ***User (Student):*** Attempts to select a table.
- ***System:*** Validates table availability.

Flow of Events:

1. ***Select a table:***
    - The user attempts to choose a table on the selection screen.
2. ***Validation check:***
    - The system checks if the selected table is already reserved.
3. ***Display warning:***
    - If the table is occupied, the system displays a message: "This table is already reserved."
4. ***Choose another table:***
    - The user selects a different table.

Entry Condition:

- The user is logged into the system and viewing the table selection screen.

Exit Conditions:

- The user selects an available table.
- The user decides to leave the table selection process.

Use Case Name: ***Reservation***
Participating Actors:

- ***User (Student):*** Requests to extend their reservation time.
- ***System:*** Updates the reservation details.

Flow of Events:

1. ***Initiate update:***
   - The user clicks on the "Update the Reservation Time" option on the main screen.
2. ***Set new duration:***
   - The user specifies the additional time they wish to reserve the table for.
3. ***Confirm update:***
   - The user confirms the update by clicking the "Update" button. The system processes the change and updates the reservation period.
4. ***Display updated duration:***
   - The system shows the new reservation time to the user.

Entry Condition:

- The user must have an active reservation in the system.

Exit Conditions:

- The reservation time is successfully updated.
- The user exits the update process without making changes.

3.3.3 Object Model

The Object Model designed for the Library Management System (LMS) reflects a user-centric approach to solving the challenges of managing shared library spaces effectively. This system leverages core object-oriented principles to provide seamless table reservations, updates, and notifications for students. Below is a detailed description of the components and their responsibilities based on the provided UML diagram:

Components and Responsibilities

1. Student Class

The Student class represents the primary system users. It contains attributes such as studentID, name, email, and password for authentication and profile management.

- **Attributes:**
    - studentID: Unique identifier for the student.
    - name: Name of the student.
    - email: Email address for communication.
    - password: Password for login authentication.
- **Methods:**
    - login(): Allows the student to log in to the system.
    - logout(): Logs the student out of the system.

2. Table Class

The Table class models the library's physical study tables.

- **Attributes:**
    - tableID: Unique identifier for each table.
    - location: Specifies the table's location in the library.
    - status: Indicates the table's current status (e.g., available, reserved, occupied).
    - capacity: The number of seats at the table.
- **Methods:**
    - reserveTable(studentID, time): Allows a table to be reserved by a student.
    - updateReservation(time): Updates the reservation details for a table.
    - checkAvailability(): Checks if the table is available for reservation.

3. Reservation Class

The Reservation class manages table reservations.

- **Attributes:**
  - reservationID: Unique identifier for each reservation.
  - tableID: Identifies the table being reserved.
  - studentID: Links the reservation to a specific student.
  - startTime: The start time of the reservation.
  - endTime: The end time of the reservation.
- **Methods:**
  - createReservation(): Creates a new reservation for a table.
  - cancelReservation(): Cancels an existing reservation.

4. System Class

The System class acts as a control class to handle core system logic.

- **Attributes:**
  - currentDateTime: Tracks the system's current date and time.
- **Methods:**
  - sendNotification(studentID): Sends notifications to students regarding their reservations.
  - performHourlyUpdate(): Updates the system and table statuses at regular intervals.

Boundary Classes

Boundary classes serve as interfaces between the user and the system.

- **LoginScreen:**
  - displayLogin(): Displays the login interface.
  - getUserCredentials(): Collects user credentials for authentication.
- **TableViewScreen:**
  - displayAvailableTables(): Displays a list of available tables.

   - selectTable(): Allows users to select a table for reservation.

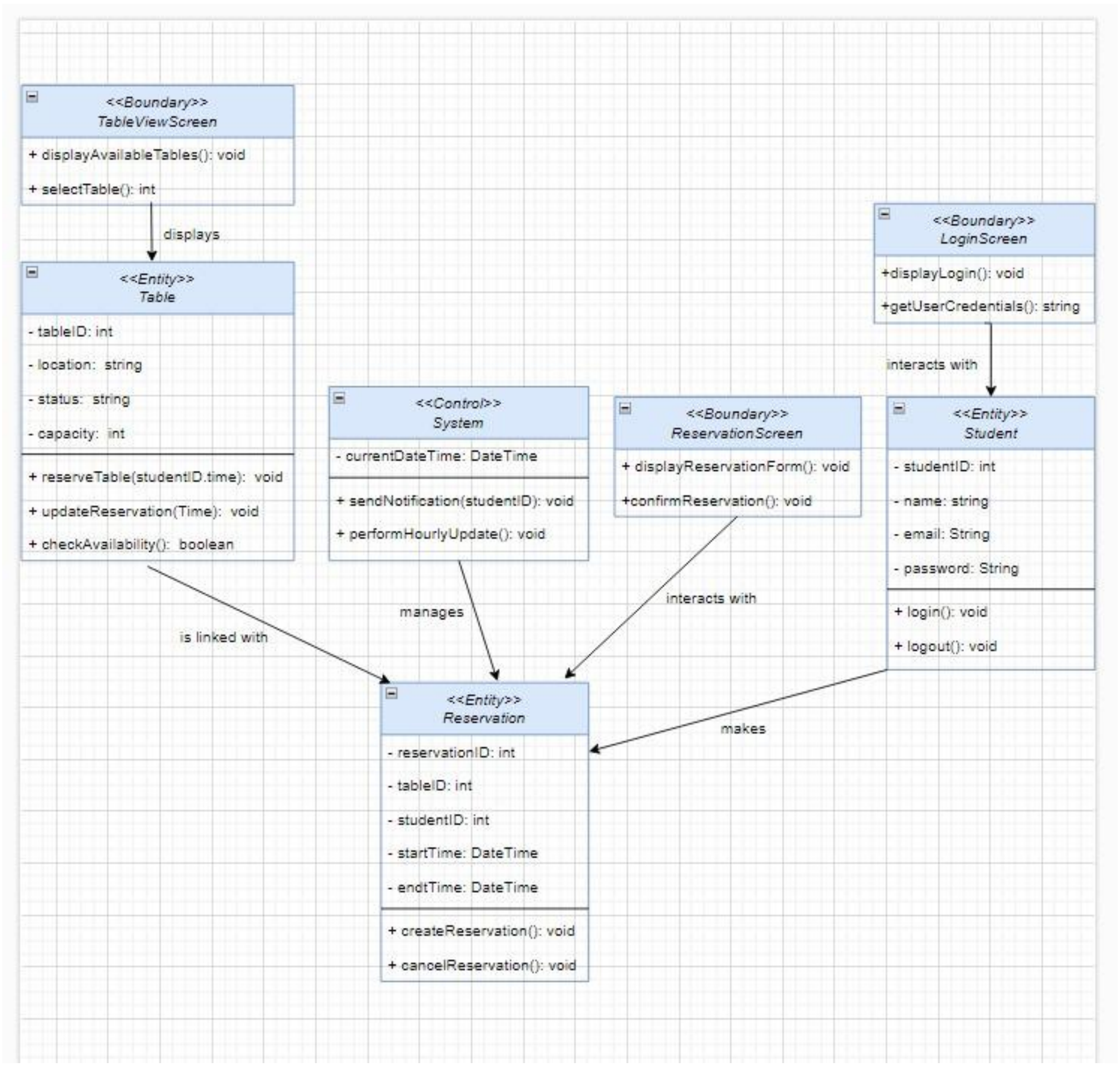
- **ReservationScreen:**
   - displayReservationForm(): Displays the form for creating or managing reservations.
   - confirmReservation(): Confirms and finalizes a reservation.

Relationships and Interactions

- **Student – Reservation Relationship:**
   - A one-to-many relationship exists since each student can make multiple reservations over time.
- **Table – Reservation Relationship:**
   - A one-to-one relationship ensures each table is linked to only one reservation during its reserved period.
- **System – Reservation Relationship:**
   - The System class manages all reservations and ensures their statuses are updated as necessary.

Key Observations from the UML Diagram

- **Entity Objects:**
   - Student, Table, Reservation
- **Boundary Objects:**
   - LoginScreen, TableViewScreen, ReservationScreen
- **Control Object:**
   - System

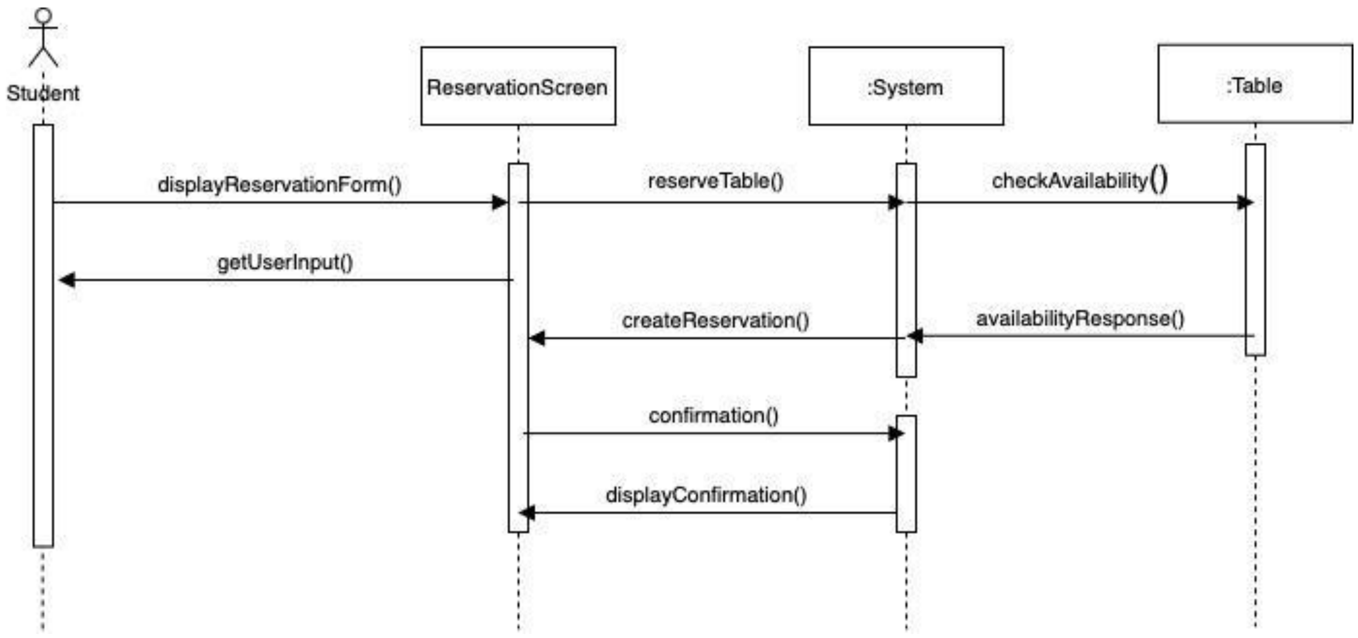


<<Boundary>>
TableViewScreen
+ displayAvailableTables(): void
+ selectTable(): int
displays
<<Entity>>
Table
- tableID: int
- location: string
- status: string
- capacity: int
+ reserveTable(studentID.time): void
+ updateReservation(Time): void
+ checkAvailability(): boolean
<<Control>>
System
- currentDateTime: DateTime
+ sendNotification(studentID): void
+ performHourlyUpdate(): void
<<Boundary>>
ReservationScreen
+ displayReservationForm(): void
+confirmReservation(): void
<<Boundary>>
LoginScreen
+displayLogin(): void
+getUserCredentials(): string
interacts with
<<Entity>>
Student
- studentID: int
- name: string
- email: String
- password: String
+ login(): void
+ logout(): void
is linked with
manages
interacts with
makes
<<Entity>>
Reservation
- reservationID: int
- tableID: int
- studentID: int
- startTime: DateTime
- endtTime: DateTime
+ createReservation(): void
+ cancelReservation(): void

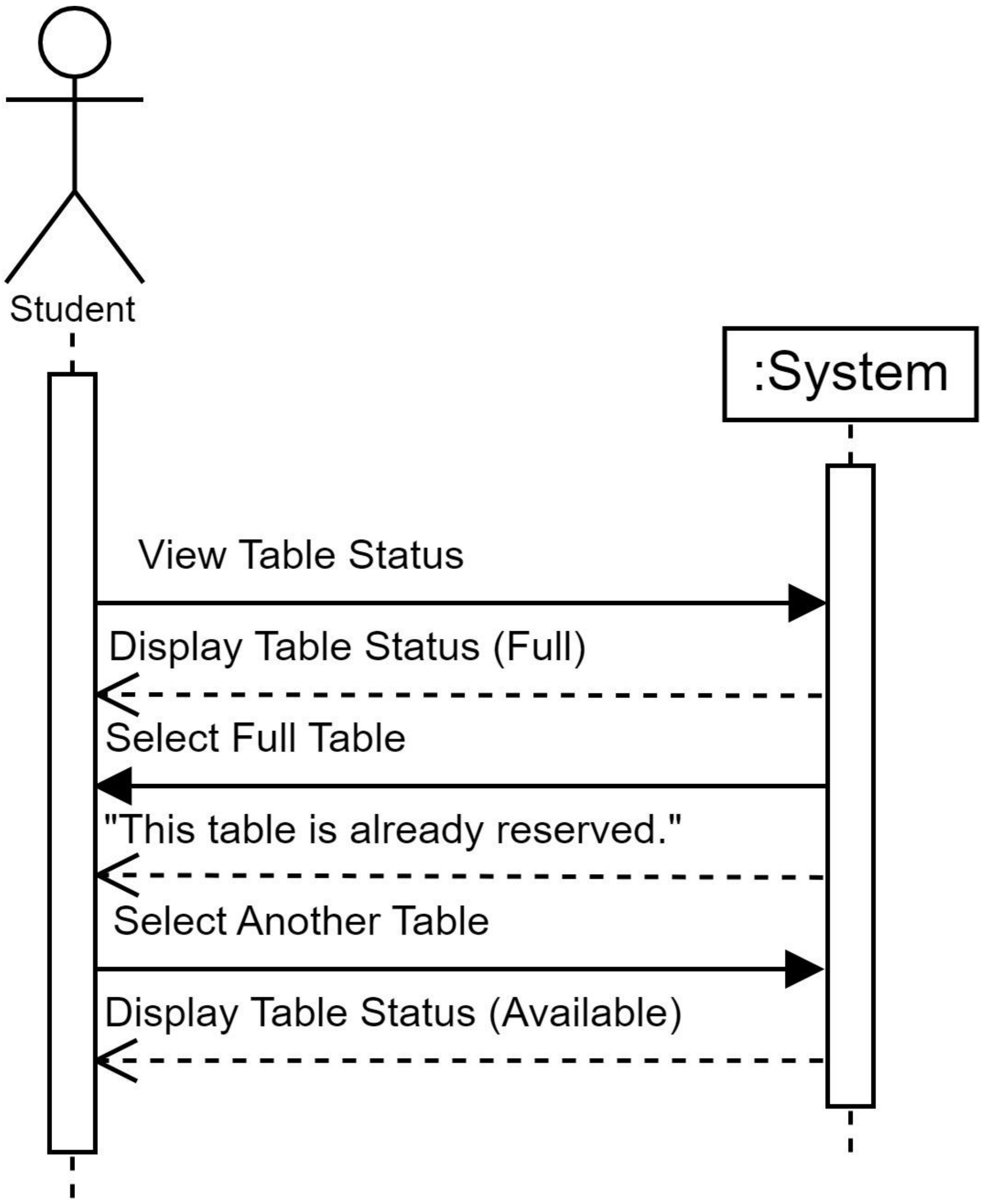
3.3.4 Dynamic Models

3.3.4.1 *Sequence Diagrams*

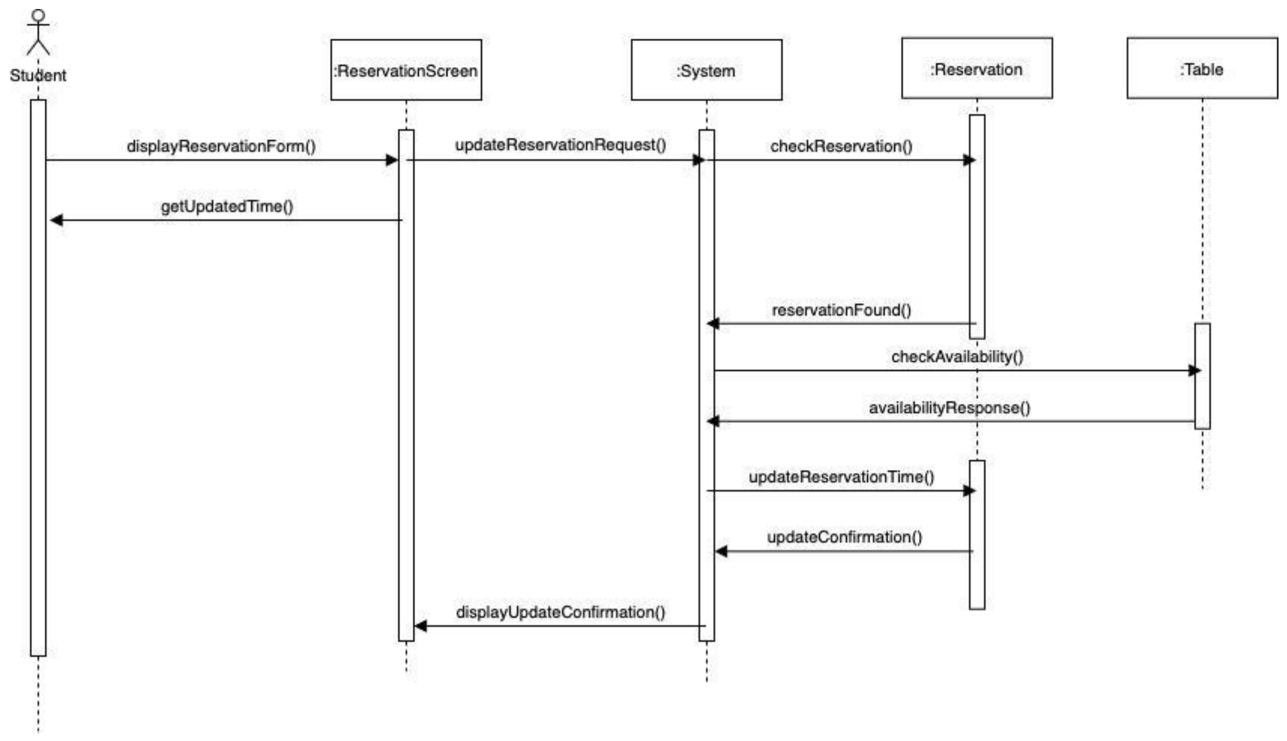
- **Table Reservation Process Scenario**



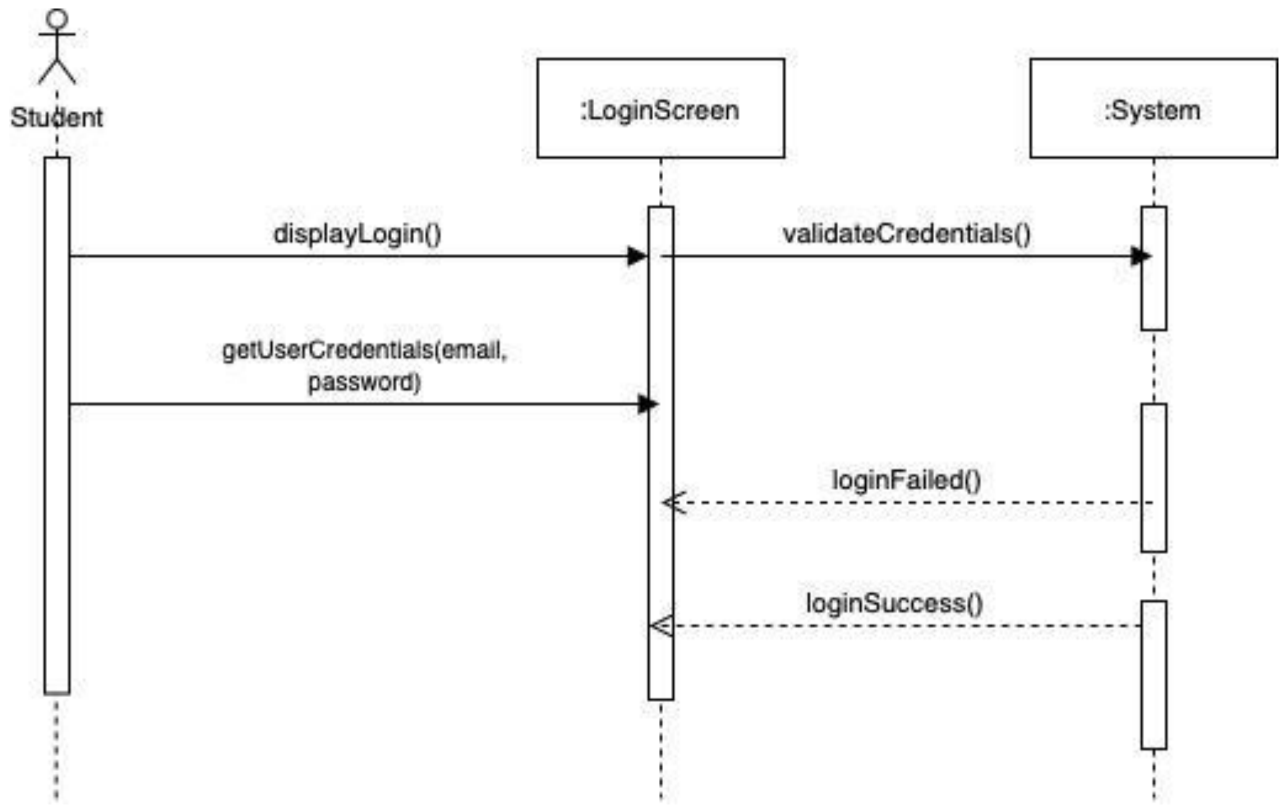
- **Booked Table Warning Scenario**


Student
:System
View Table Status
Display Table Status (Full)
Select Full Table
"This table is already reserved."
Select Another Table
Display Table Status (Available)

- **Update Reservation Scenario**

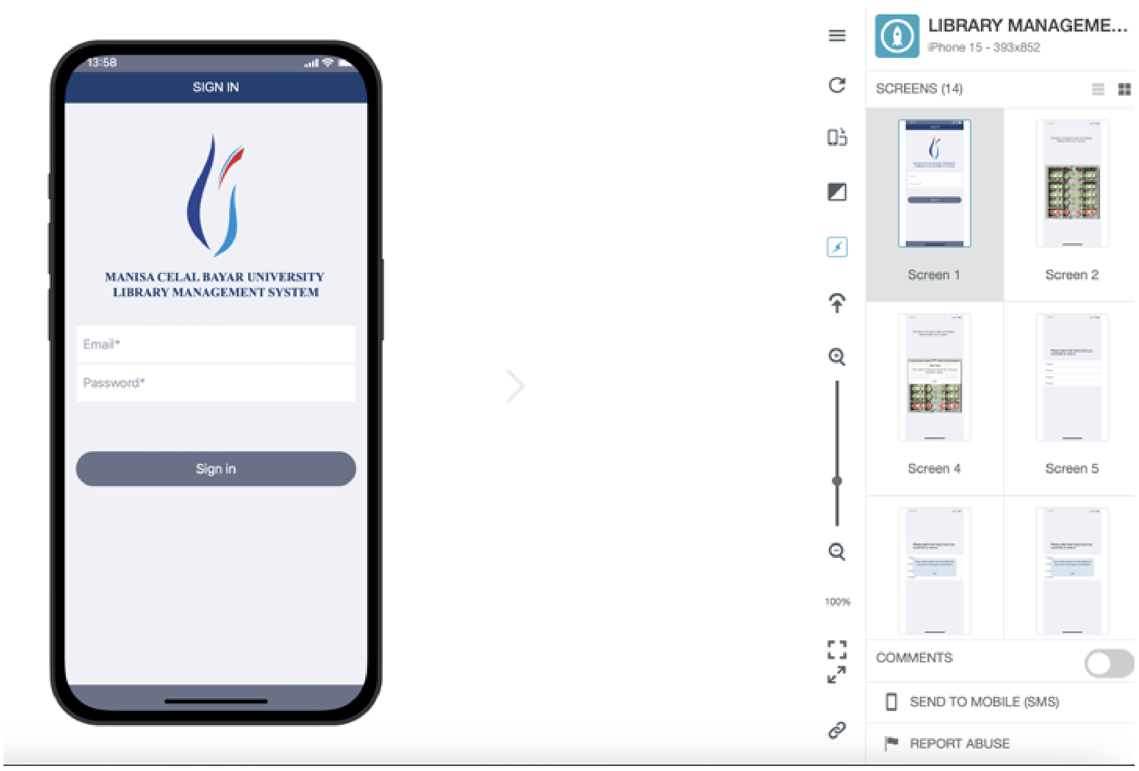


- **Login Scenario**

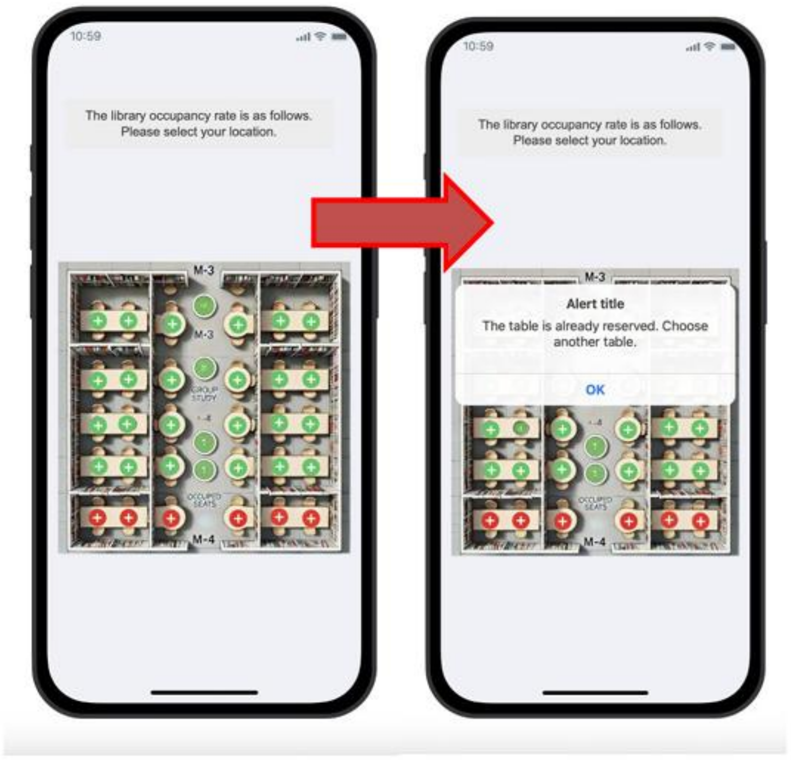


3.3.5 User Interface Mock-ups

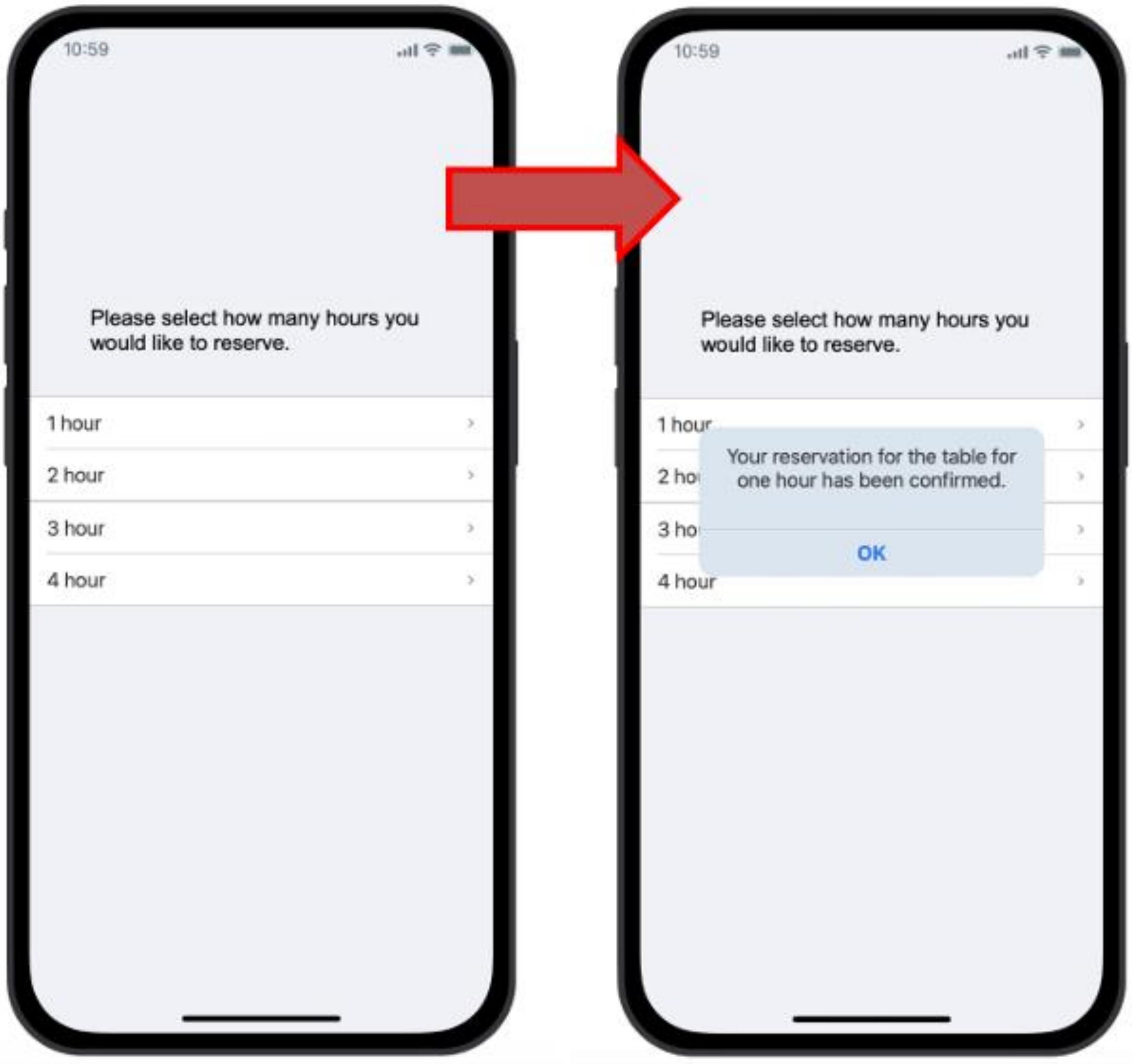
Working UI design link: https://pr.to/29G5OK/



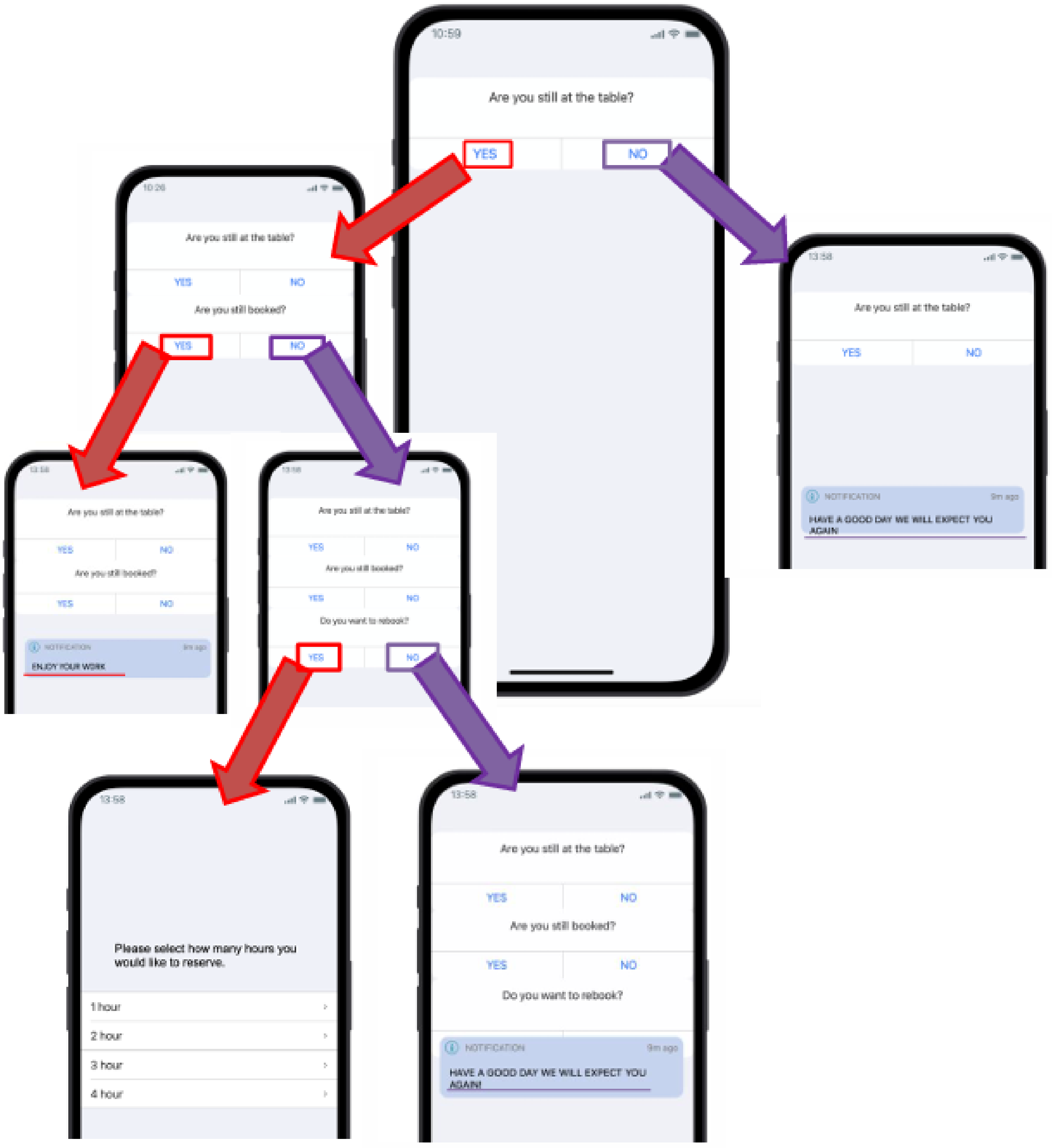
This screen pops up when you try to select a previously selected table:


10:59
The library occupancy rate is as follows. Please select your location.
M-3
GROUP STUDY
OCCUPIED SEATS
M-4
10:59
The library occupancy rate is as follows. Please select your location.
Alert title
The table is already reserved. Choose another table.
OK

This screen pops up when the selection is confirmed. It also includes information about the time you selected:


Please select how many hours you would like to reserve.

Your reservation for the table for one hour has been confirmed.



10:59

Are you still at the table?

10:26

Are you still at the table?

Are you still booked?

13:58

Are you still at the table?

NOTIFICATION 9m ago

HAVE A GOOD DAY WE WILL EXPECT YOU AGAIN!

13:58

Are you still at the table?

Are you still booked?

NOTIFICATION 9m ago

ENJOY YOUR WORK

13:58

Are you still at the table?

Are you still booked?

Do you want to rebook?

13:58

Please select how many hours you would like to reserve.

1 hour
2 hour
3 hour
4 hour

13:58

Are you still at the table?

Are you still booked?

Do you want to rebook?

NOTIFICATION 9m ago

HAVE A GOOD DAY WE WILL EXPECT YOU AGAIN!

4 Glossary

1. **Library Management System (LMS):** A software system developed to manage the library's occupancy status, table reservations, and user entry and exit.

2. **User Login:** Module that allows users (students) to log in to the system. This module authenticates with a username and password.

3. **Empty Tables:** Available empty tables in the library. Available tables that students can make reservations for, displayed instantly in the system.

4. **Occupied Tables:** Tables that have been reserved by other users in the library or are not available for use.

5. **Table Reservation:** System feature that allows students to reserve a table for a certain period of time. This process makes the table selected by the student appear “reserved” for other users.

6. **Reservation Time Update:** Feature that allows students to extend their current reservation time. If the student wants to continue using the table, they can update the reservation time.

7. **Hourly Updates:** Updates where the system checks every hour to see if the table is still reserved. This process verifies whether the student is at the table.

8. **Reservation Expiration Notification:** A warning message that informs students that their reservation period has expired. This feature reminds students that they need to vacate the table or extend it.

9. **Performance:** The ability of the system to instantly update information such as table status and to perform user operations quickly.

10. **Reliability:** The ability of the system to work without interruption and perform user operations correctly. The system should quickly provide solutions in case of errors.

11. **Usability:** The system should be user-friendly, allowing users to find tables quickly and easily and make reservations.

12. **Supportability:** The system should be manageable in terms of maintenance and updates. There is a support desk where users can get help with problems they encounter.

5. Appendix

- Annex – I: Distribution of Work
- Annex – II: Meeting Minutes

Distribution of Work

As a team, we designed and implemented the system collaboratively, dividing tasks based on individual responsibilities and strengths.

Ertuğrul was responsible for creating the object model, ensuring that all system entities and their relationships were clearly defined. Umut collaborated with Ertuğrul to design the use case model, providing a comprehensive representation of system functionalities and user interactions. Umut contributed by explaining the concepts and details of the object model to the team, fostering a shared understanding of its structure and purpose.

For the dynamic model, both Sena and Aslı worked together to create the sequence diagrams, illustrating the interactions between the system components. Aslı also ensured the dynamic model's accuracy by carefully reviewing and verifying the sequence flows.

Sıla focused on the table of contents and performed a thorough review of the report to ensure its completeness and coherence.

Finally, Sena took the lead in writing the report, consolidating all the work and documenting the system design, workflows, and analyses in a clear and professional manner.

Meeting Minutes

Date:	08.10.2024
Location:	Discord
Duration:	1 hour
Participants:	Sıla Naz ASLAN Sena KILINÇ Aslı ÇELİK Umut AKBAŞ Zafer SARIK İsmail Ertuğrul BOZKURT
Content of the meeting (briefly explain the agenda, decisions, work distributions, etc.)	
Topic has been determined.	

Date:	29.10.2024
Location:	MS Teams
Duration:	2 hours (18.00-20.00)
Participants:	Sıla Naz ASLAN Sena KILINÇ Aslı ÇELİK Umut AKBAŞ Zafer SARIK İsmail Ertuğrul BOZKURT
Content of the meeting (briefly explain the agenda, decisions, work distributions, etc.)	
As for the content of the meeting, we held a meeting to evaluate the general progress of the assignment, determine the distribution of tasks and clarify the responsibilities of each team member. First, a brief assessment was made of the scope and main goals of the project. At the end of the meeting, each member clarified the tasks related to their area of responsibility and a work plan was created to take the necessary steps for the successful completion of the project.	

Date:	17.11.2024
Location:	MS Teams
Duration:	2 hours (15.00-17.00) and 2 Hours (18.00-20.00)
Participants:	Sıla Naz ASLAN Sena KILINÇ Aslı ÇELİK Umut AKBAŞ Zafer SARIK İsmail Ertuğrul BOZKURT
Content of the meeting (briefly explain the agenda, decisions, work distributions, etc.)	
During the meeting, the team discussed the assignment details, clarified the objectives, and outlined the scope of the project. Ertuğrul took the lead in analyzing the requirements and guided the team to ensure efficient progress. Task allocation was finalized: Zafer and Sıla was assigned to research suitable platforms, recommending Proto.io as the most effective tool for designing the user interface mockup. Umut, Sena, and Aslı collaborated to create the mockup, focusing on usability and aesthetics. Meanwhile, Aslı was tasked with drafting the project report, consolidating all the work and results. The team reviewed progress, shared initial ideas for the mockup design, and aligned on the timeline for completing their tasks. It was agreed that everyone would provide feedback on the mockup and report in the next meeting to finalize the project.	

Date:	08.12.2024
Location:	MS Teams
Duration:	2 hours (20.00-22.00) and 1 hour (22.15-23.15)
Participants:	Sıla Naz ASLAN Sena KILINÇ Aslı ÇELİK Umut AKBAŞ İsmail Ertuğrul BOZKURT

Content of the meeting (briefly explain the agenda, decisions, work distributions, etc.)

As a team, we designed and implemented the system collaboratively, dividing tasks based on individual responsibilities and strengths.

Ertuğrul was responsible for creating the object model, ensuring that all system entities and their relationships were clearly defined. Umut collaborated with Ertuğrul to design the use case model, providing a comprehensive representation of system functionalities and user interactions. Umut contributed by explaining the concepts and details of the object model to the team, fostering a shared understanding of its structure and purpose.

For the dynamic model, both Sena and Aslı worked together to create the sequence diagrams, illustrating the interactions between the system components. Aslı also ensured the dynamic model's accuracy by carefully reviewing and verifying the sequence flows.

Sıla focused on the table of contents and performed a thorough review of the report to ensure its completeness and coherence. Finally, Sena took the lead in writing the report, consolidating all the work and documenting the system design, workflows, and analyses in a clear and professional manner.